A Project Report

on

Comparison of various algorithms for fake review detection on E-Commerce websites

submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

By

19WH1A0579 Ms. G. SUVIDHA REDDY

19WH1A0568 Ms. K. AKSHAYA

19WH1A0563 Ms. B. AKSHITHA

Under the esteemed guidance of

Dr. Reya Sharma

Assistant Professor, CSE

Department of Computer Science Engineering

BVRIT HYDERABAD

College of Engineering for Women

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2022-2023

DECLARATION

We hereby declare that the work described in this report, entitled **"COMPARISON OF VARIOUS ALGORITHMS FOR FAKE REVIEW DETECTION ON E-COMMERCE WEBSITES"** which is submitted by us in partial fulfilment for the award ofthe degree of **Bachelor of Technology** in the department of **Computer Science Engineering** at **BVRIT HYDERABAD College of Engineering for Women**, affiliated to **Jawaharlal Nehru Technological University Hyderabad**, Kukatpally, Hyderabad – 500085 is the result of original work carried out under the guidance of **Dr. Reya Sharma, Assistant Professor, Department of CSE.**

This work has not been submitted for any Degree / Diploma of this or any other institute/university to the best of our knowledge and belief.

Sign with Date
Ms. G. Suvidha Reddy
(19WH1A0579)

Sign with Date
Ms.K.Akshaya
(19WH1A0568)

Sign with Date
Ms. B. Akshitha
(19WH1A0563)

Department of Computer Science Engineering

BVRIT HYDERABAD
College of Engineering for Women

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

CERTIFICATE

This is to certify that the project work report, entitled **"Comparison of various algorithms for fake review detection on E-Commerce websites"** is a bonafide work carried out by **Ms. G. Suvidha Reddy (19WH1A0579), Ms. K. Akshaya (19WH1A0568), Ms. B. Akshitha (19WH1A0563)** in partial fulfillment for the award of B.Tech degree in **Computer Science Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to **Jawaharlal Nehru Technological University Hyderabad, Hyderabad** under my guidance and supervision.

The results embodied in the project work have not been submitted to any other university or institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. E. Venkateswara Reddy	**Dr. Reya Sharma**
Professor and HoD,	**Assistant Professor**
Department of CSE	**Department of CSE**

External Examiner

ACKNOWLEDGEMENTS

The satisfaction that accompanies in successful completion of the task would be incomplete without the mention of the people who made it possible.

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women**, for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. E. Venkateswara Reddy, Head, Department of CSE, BVRIT HYDERABAD College of Engineering for Women**, for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Dr. Reya Sharma, Assistant Professor, CSE, BVRIT HYDERABAD College of Engineering for Women**, for her constant guidance and encouragement throughout the project.

Finally, we would like to thank our Major Project Coordinator, all Faculty and Staff of CSE department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents** and **Friends** for giving moral strength and constant encouragement.

G. Suvidha Reddy (19WH1A0579)

K. Akshaya (19WH1A0568)

B. Akshitha (19WH1A0563)

ABSTRACT

People frequently purchase items online, and many of e-commerce sites provide a review option. Some website owners may employ spammers to write false reviews or use machine generated fake reviews in order to boost product sales and also decrease the sales of their competitors. Many approaches have been proposed by researchers in the past. We aim to find the best way for the detection of span reviews through NLP and DEEP LEARNING models and deduce the best way out of all.

Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

BERT	Bidirectional Encoder Representations from trasformers
ROBERTa	Robustly optimized Bert pretraining approach
LSTM	Long short-term memory
BiLSTM	Bidirectional Long short memory
RNN	Recurrent neural networks
SVM	Support Vector Machine
KNN	K Nearest Neighbor

1. INTRODUCTION

Internet is playing an essential role for modern information systems. Applications, such as e-commerce websites, are becoming popularly available for people to purchase different types of products online. During such an online purchasing process, users often rely on online review reports from previous customers to make the final decision. Because online reviews are playing essential roles for the selling of online products, some vendors are providing fake/spam reviews to mislead the customers. Any false reviews of the products may result in unfair market competition and financial loss for the customers or vendors.

In this research, we aim to distinguish between spam and non-spam reviews by using DEEP LEARNING and NLP models. There are two types of datasets commonly used in this research area: real fake and real-life reviews of various purchasing sites from Kaggle. The algorithms used in the process are LSTM, ROBERTa, BiLSTM and NBLR. The approach follows 4 main steps: (1) Collection of datasets. (2) Evaluate the sample reviews using quantitative metrics and qualitative assessment. (3) Train classifier algorithms to detect artificially generated and fake reviews from the real ones. (4) Compare the accuracy of various classification algorithms via statistical testing

1.1 Problem Statement:

In this research, we aim to compare various Machine Learning models and deduce an approach to differentiate between spam and non-spam reviews with a higher accuracy so that it can be used to real life e-commerce websites.

1.2 Objective:

The techniques for detecting fake reviews have extensively been explored in the past twelve years. However, there still lacks a survey that can analyze and summarize the existing approaches. To bridge up the issue, this research details the task of fake review detection, summing up the existing best models and methods.

1.3 Proposed System:

This system compares and analyzes Roberta, LSTM, BiLSTM and NBLR algorithms on the amazon fake and real dataset of top 10 products and decide the best way to classify our reviews.

The algorithms which we have selected have better performance than supervised machine learning algorithms like KNN, random forest, SVM for text classification applications which we got to know through our literature survey.

2. LITERATURE SURVEY

Title: Creating and detecting fake reviews of online products [1]

Authors: Joni Salminen, Chandrashekhar Kandpal, Ahmed Mohamed Kamel,

Soon-gyo Jung, Bernard J. Jansen

Published year: 2022

Summary: Detection of fake reviews is a problem for researchers, e-commerce sites, and firms engaged in online business. The results indicate that current text generation methods yield fake reviews that appear so realistic that it is challenging for a human to detect them. Reviews of amazon website were used and the algorithms used are Open AI(Open Artificial Intelligence), NBSVM(Support Vector Machine with Naïve Bayes features) and ROBERTa. ROBERTa has given the maximum accuracy out of all.

Numericals: Using this Roberta algorithms accuracy is 96.64%.

Title: Detecting and Analysing Fake Opinions Using Artificial Intelligence Algorithms [4]

Authors: Fawaz Waselellah Alsaade, Mosleh Hmoud Al-Adhaileh

Published year: 2021

Summary: In this paper, Yelp product reviews were used to conduct empirical tests of the performance of two neural network models, CNN and BiLSTM, for identifying fake opinions. Based on the learning word embeddings of the text of reviews, the proposed models were used to classify those product reviews as fake or truthful. By comparing the results obtained from several experiments, we found that the BiLSTM model provides higher performance than the CNN model. In text classification tasks, neural networks can appropriately catch global semantic information over sentence vectors. As a result, deep learning-based models outperform the baseline models in terms of accuracy. The experimental results also showed that the CNN model was superior to the BiLSTM model in data processing time.

Numericals: The model works with 78% accuracy. Their experimental result was an

F1 score of 90% with BiLSTM.

Title: Fake online review recognition algorithm and optimisation research based on deep learning[5]

Authors: Jiani Hou, Aimin Zhu

Published year: 2021

Summary: This research suggests a hybrid neural network model that combines engineering traits. It is contrasted with the convolutional neural network model, the LSTM model, the bilateral LSTM model, and the hybrid model, which combines the convolutional and recurrent neural networks using three different approaches to extract global features. The effectiveness of recognition is assessed based on the assessment index. The aforementioned models optimize and make use of the word vectors with random initialization. The experimental findings demonstrate that the local and global properties of text can be improved more effectively when convolutional and recurrent neural networks are combined. By adding feature engineering, it is possible to get rid of some of the comments made by erroneous reviewers and improve the model's ability to recognize objects.

Numericals: accuracy obtained is 91.5% using the BiLSTM

Title: Detection of Fake User Reviews with Deep Learning[3]

Authors: Kenan Tasagal**,** Özlem Uçar

Published:2018

Summary: In our research,LSTM, BILSTM, and CNN-LSTM accuracy rates for identifying incorrect interpretations were evaluated.Models from deep learning models and attempted to determine which of these 3 models was the most successful. After the experiments, we discovered that the CNN-LSTM model from the LSTM, BILLSTM, and CNN-LSTM models produced superior results than other models with an
accuracy of 87%.

Numericals: accuracy of 87% was obtained with CNN-LSTM

Title: Natural Language Processing based Online Fake News Detection Challenges – A Detailed Review[10]

Authors: Vaishali Vaibhav Hirlekar, Dr.Arun Kumar

Published year: 2020

Summary: The goal of this study was to review, synthesize, compare, and assess the most recent research on fake news. It includes identifying and intervening strategies, as well as quantitative and qualitative analysis of fake news. As we said, the machine learning answer to the issue of false news, rumors, and disinformation detection is called fake news detection. The composite classification system, in particular, consists of neural networks made of traditional classification algorithms that primarily use lexical analysis of the objects as the primary characteristic for prediction and usage of outside background data

Numericals: Accuracy observed in the LSTM is 66 %, for GRU is 66.4 %, for RNN is 71.0 %, for RF is 70 % and for LR is 66.9 %

Title: Ensemble Learning for Detecting Fake Reviews [7]

Authors: Luis Gutierrez-Espinoza1, Faranak Abri1, Akbar Siami Namin, Keith S. Jones, David R. W. Sears

Summary: In this work, novel dataset of fake reviews, along with the results of the binary classification using machine/deep learning techniques were presented. Additionally, they applied ensemble learning-based approaches using Random Forest, bagging, and adaboost ensembles, with SVMs and MLPs as weak classifiers with optimized hyperparameters .The results show that, using document embedding from Doc2Vec and after hyperparameter optimization, stand-alone classifiers achieve up to 68.2% accuracy in the case of MLP. Ensemble learning-based classifiers achieve up to 77.3% accuracy with the adaboost ensemble of MLPs. In every case, the ensemble of classifiers outperforms their respective base classifier, either Random Forest and XGBT with Decision Tree, or the bagging/adaboost ensembles with their respective SVMs or MLPs. Regarding the ensemble approach, adaboost seems to produce the most consistent results.

Numericals: Maximum accuracy was 79.1% and 76.9% precision with decision tree

Title: Fake review detection on yelp dataset using classification techniques in machine learning [8]

Authors: Andre Sihombing and A.C.M. Fong

Summary: This paper has reviewed four popular ML algorithms for finding fake Yelp reviews.The experiments showed very high score in prediction, when using XGBoost.

Imbalance in the dataset nneds to be handled for sure.SVM gave the best results but took l ongest time to train.Gaussian Naïve Bayes gave the lowest score.

Numericals: Using XGBoost Precision was 100% nad F1 score was 99%.

Title: Survey of review spam detection using machine learning techniques.[2]

Authors: Michael Crawford, Taghi M. Khoshgoftaar, Joseph D. Prusa

Summary: By extracting meaningful features from the text using Natural Language Processing (NLP), it is possible to conduct review spam detection using various machine learning techniques. Additionally, reviewer information, apart from the text itself, can be used to aid in this process. In this paper, we survey the prominent machine learning techniques that have been proposed to solve the problem of review spam detection and the performance of different approaches for classification and detection of review spam. The majority of current research has focused on supervised learning methods, which require labeled data, a scarcity when it comes to online review spam. Research on methods for Big Data are of interest, since there are millions of online reviews, with many more being generated daily

Numericals: The results of a 5-fold cross validation experiment with an SVM classifier using bigram and POS features resulted in an accuracy of 68.1 % for the real-world fake reviews. This is far lower than the 90 % reported by Ott et al.

Title: A study of machine learning classifiers for spam detection [5]

Authors: Sharawan Kumar Trivedi

Summary: In the present world, there is a need of emails communication but unsolicited emails hamper such communications. The present research emphasises to build a spam classification model with/without the use of ensemble of classifiers methods have been incorporated. Through this study, the aim is to distinguish between ham emails and spam emails by making an efficient and sensitive classification model that gives good accuracy with low false positive rate. Greedy Stepwise feature

search method has been incorporated for searching informative feature of the Enron email dataset. The comparison has been done

among different machine learning classifiers (such as Bayesian, Naïve.

Bayes, SVM (support vector machine), J48 (decision tree), Bayesian with Adaboost, Naïve Bayes with Adaboost). The concerned classifiers are tested and evaluated on metric (such as F-measure (accuracy), False Positive Rate, and training time).

Numericals: By analysing all these aspects in their entirety, it has been found that SVM is the best classifier to be used. It has the high accuracy and the low false positive rate. However, training time of SVM to build the model is high, but as the results on other parameters are positive, the time does not pose such an issue.

3. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

3.1 Requirement Gathering :

Software Requirements:

- **Language Used:** Python 3.8
- **Operating System:** Windows
- **Software:** Anaconda, Jupyter notebook

Hardware Requirements:

- **Processor:** Intel Core i3 (Minimum)
- **RAM:** 4GB (Minimum)

3.2 Technologies Description:

Python:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

Jupyter Notebook:

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Pytorch:

PyTorch is a fully featured framework for building deep learning models, which is a type of machine learning that's commonly used in applications like image recognition and language processing. Written in Python, it's relatively easy for most machine learning developers to learn and use.

Anaconda:

Anaconda distribution is a free and open-source platform for Python/R programming languages. It can be easily installed on any OS such as Windows, Linux, and MAC OS. It provides more than 1500 Python/R data science packages which are suitable for developing machine learning and deep learning models.

Anaconda distribution provides installation of Python with various IDE's such as Jupyter Notebook, Spyder, Anaconda prompt, etc. Hence it is a very convenient packaged solution which you can easily download and install in your computer. It will automatically install Python and some basic IDEs and libraries with it.

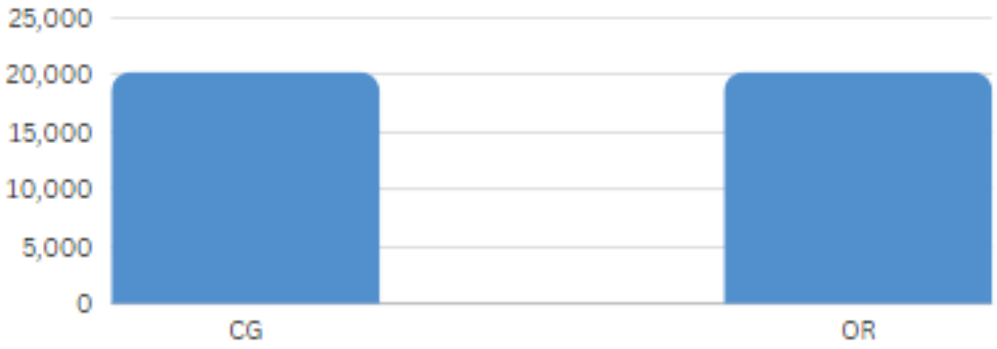
4. METHODOLOGY

4.1 Datasets:

- Fake-Reviews-Detection/fake reviews dataset.csv at main · SayamAlt/Fake-Reviews-Detection · GitHub

Four Columns : Category, rating, label, text_

40433 rows : 20214 Real, 20213 Fake



Memory : 14,966KB

This dataset contains real amazon product reviews and Machine made reviews which are generated using UMLfit and GPT-2.

The dataset is balanced before fitting of models.

4.2 Algorithms:

Roberta: (Robustly Optimized BERT Pretrained Approach)

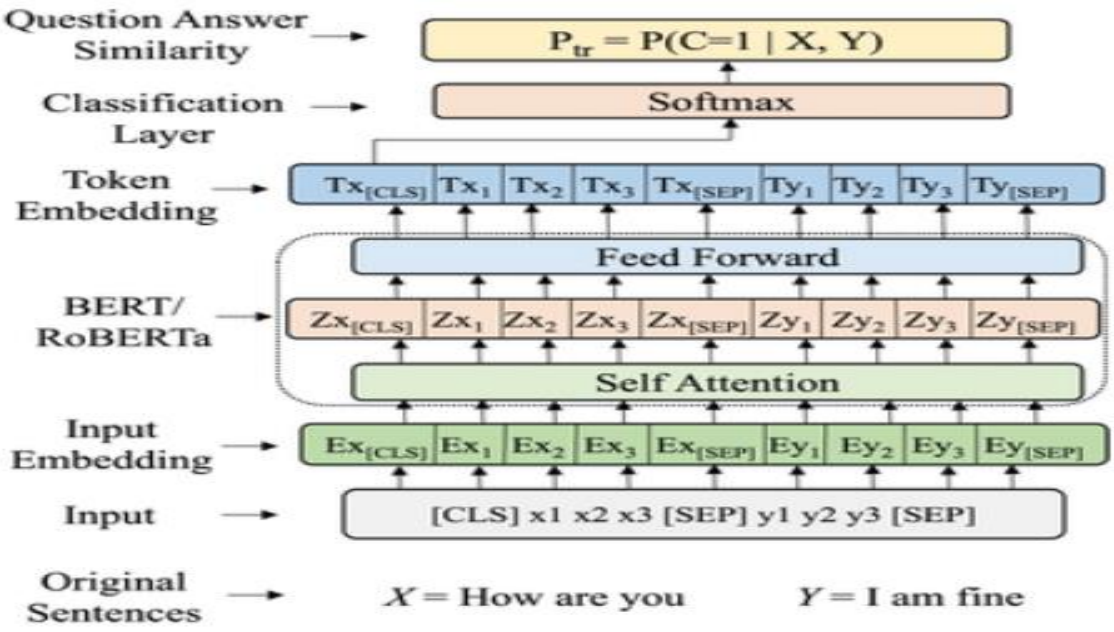


Fig 1 : Roberta Methodology

RoBERTa builds on BERT's language masking strategy, wherein the system learns to predict intentionally hidden sections of text within otherwise unannotated language examples. RoBERTa, which was implemented in PyTorch, modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance. We also explore training RoBERTa on an order of magnitude more data than BERT, for a longer amount of time.

One key difference between RoBERTa and BERT is that RoBERTa was trained on a much larger dataset and using a more effective training procedure. In particular, RoBERTa was trained on a dataset of 160GB of text, which is more than 10 times larger than the dataset used to train BERT. Additionally, RoBERTa uses a dynamic masking technique during training that helps the model learn more robust and generalizable representations of words.

RoBERTa has been shown to outperform BERT and other state-of-the-art models on a variety of natural language processing tasks, including language translation, text classification, and question answering. It has also been used as a base model for many other successful NLP models and has

become a popular choice for research and industry applications. Overall, RoBERTa is a powerful and effective language model that has made significant contributions to the field of NLP and has helped to drive progress in a wide range of applications.

LSTM (Long Short-Term Memory)

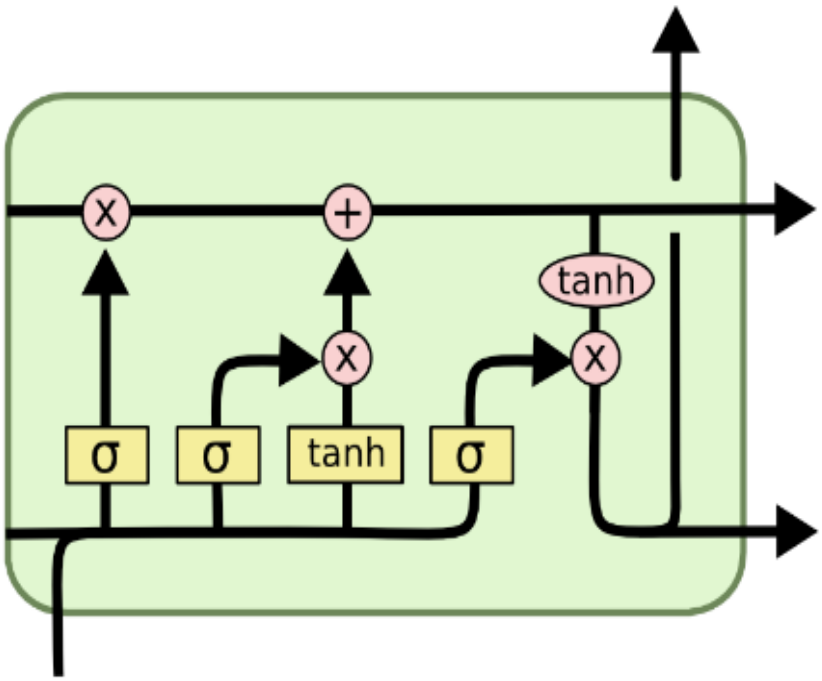


Fig 2 : LSTM Methodology

Long Short-Term Memory Networks is a deep learning, sequential neural network that allows information to persist. It is a special type of Recurrent Neural Network which is capable of handling the vanishing gradient problem faced by RNN. LSTM was designed by Hochreiter and Schmidhuber that resolves the problem caused by traditional rnns and machine learning algorithms. LSTM can be implemented in Python using the Keras library.

Let's say while watching a video, you remember the previous scene, or while reading a book, you know what happened in the earlier chapter. RNNs work similarly; they remember the previous information and use it for processing the current input. The shortcoming of RNN is they cannot remember long-term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

LSTM networks are indeed an improvement over RNNs as they can achieve whatever RNNs might achieve with much better finesse. As intimidating as it can be, LSTMs do provide better results and are truly a big step in Deep Learning. With more such technologies coming up, you can expect to

get more accurate predictions and have a better understanding of what choices to make.

BiLSTM (Bidirectional Long Short-Term Memory)

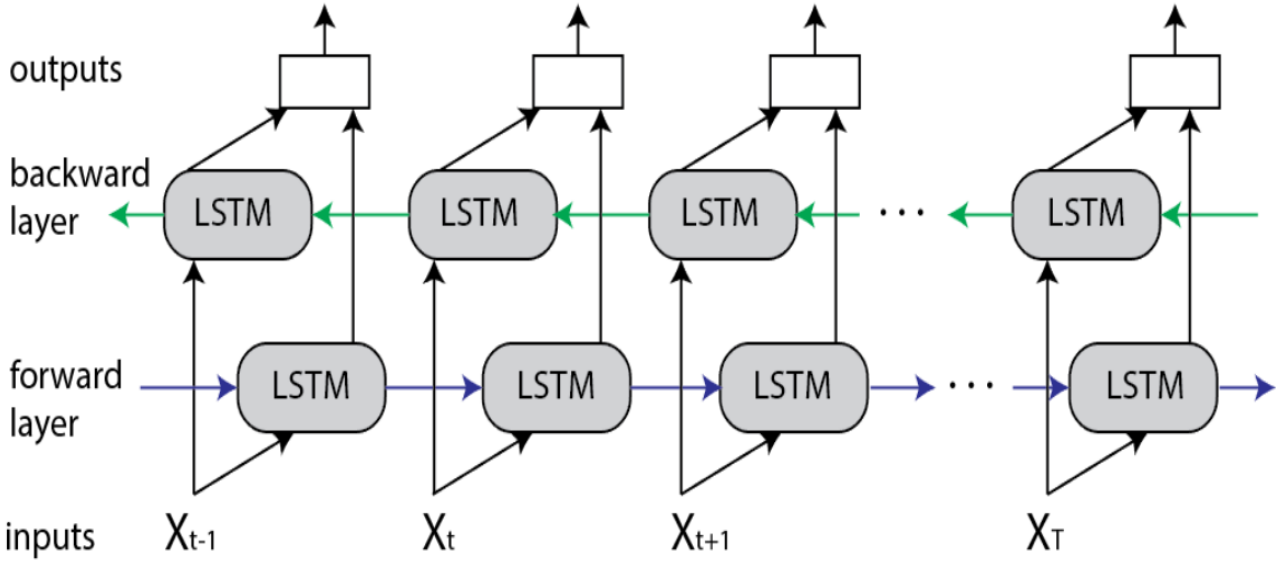


Fig 3 : Bi-LSTM Methodology

Bidirectional long-short term memory(bi-lstm) is the process of making any neural network o have the sequence information in both directions backwards (future to past) or forward(past to future). In bidirectional, our input flows in two directions, making a bi-lstm different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backwards or forward. However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information. For a better explanation, let's have an example.

In the sentence "boys go to ….." we cannot fill the blank space. Still, when we have a future sentence "boys come out of school", we can easily predict the past blank space the similar thing we want to perform by our model and bidirectional LSTM allows the neural network to perform this.

In the diagram, we can see the flow of information from backward and forward layers. BI-LSTM is usually employed where the sequence to sequence tasks are needed. This kind of network can be used in text classification, speech recognition and forecasting models.

This type of architecture has many advantages in real-world problems, especially in NLP**.** The main reason is that every component of an input sequence has information from both the past and present. For this reason, BiLSTM can produce a more meaningful output, combining LSTM layers from both directions.

NB Logistic Regression

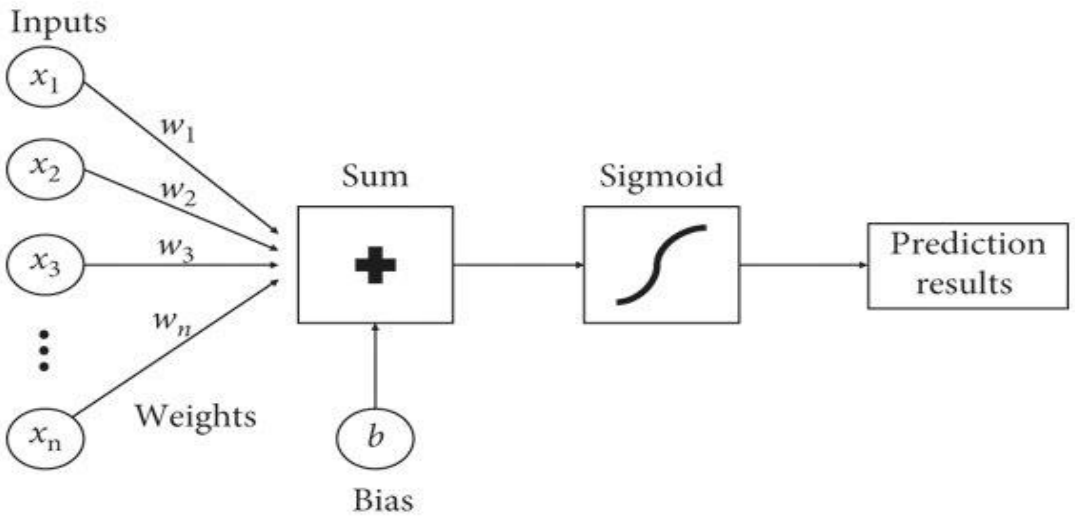


Fig 4 : NB -Logistic regression Methodology

It’s a combining model of Naïve Bayes and Logistic regression which uses basic naïve bayes feature in logistic regression to Improve the robustness and accuracy for better prediction.

5. DESIGN

Introduction:

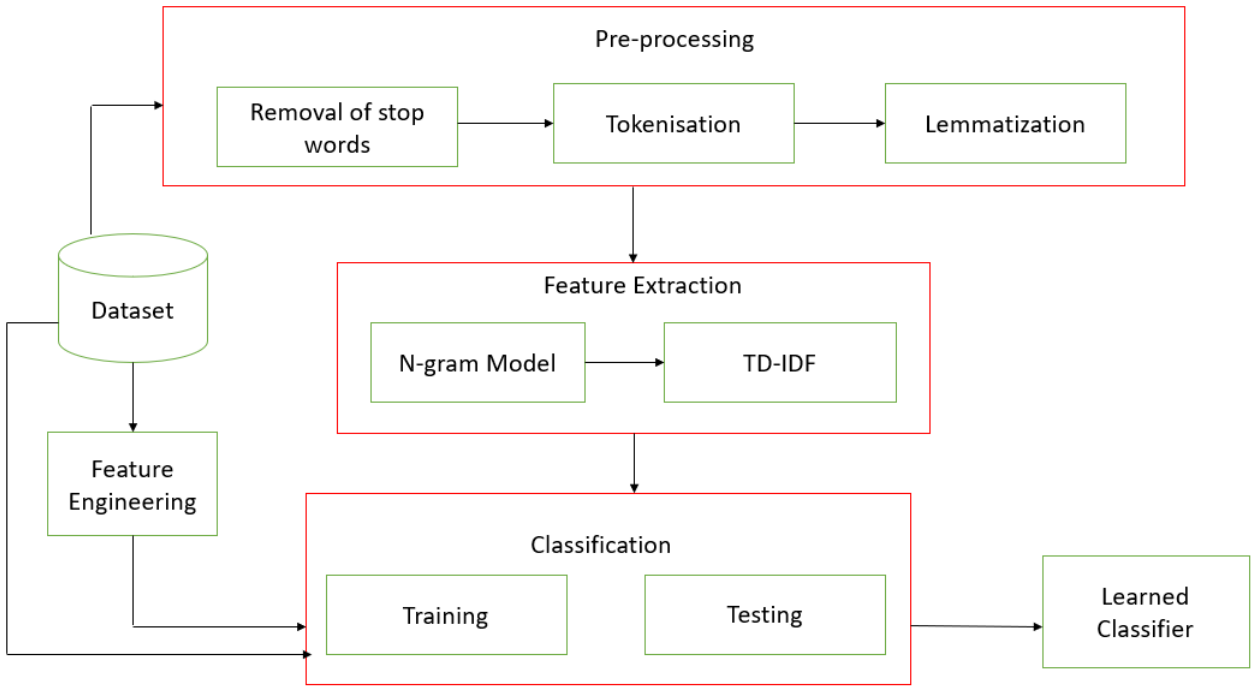


Figure 5 : Architecture Design

5.1Data Preprocessing:

The first step in the proposed approach is data preprocessing, one of the essential steps in machine learning approaches. Data preprocessing is a critical activity as the world data is never appropriate to be used. A sequence of preprocessing steps have been used in this work to prepare the raw data of the dataset for computational activities.

This can be summarized as follows:

1) Tokenization: Tokenization is one of the most common natural language processing techniques. It is a basic step before applying any other preprocessing techniques. The text is divided into individual words called tokens. For example, if we have a sentence (“wearing helmets is a must for pedal cyclists”), tokenization will divide it into the following tokens (“wearing”, “helmets”, “is”, “a”, “must”, “for, “pedal”, “cyclists”).

2) Stop Words Cleaning: Stop words are the words which are used the most yet they hold no value. Common examples of the stop words are (an, a, the, this). In this paper, all data are

cleaned from stop words before going forward in the fake reviews detection process.

3) Lemmatization: Lemmatization method is used to convert the plural format to a singular one. It is aiming to remove inflectional endings only and to return the base or dictionary form of the word. For example: converting the word (“plays”) to (“play”).

5.2 Feature Extraction:

Feature extraction is a step which aims to increase the performance either for a pattern recognition or machine learning system. Feature extraction represents a reduction phase of the data to its important features which yields in feeding machine and deep learning models with more valuable data. It is mainly a procedure of removing the unneeded attributes from data that may actually reduce the accuracy of the model.

5.3 Feature Extraction of Characters:

Fake reviews are known to have other descriptive features related to behaviors of the reviewers during writing their reviews. In this paper, we consider some of these features and their impact on the performance of the fake reviews detection process. We consider caps-count, punct-count, and emojis behavioral features. caps-count represents the total capital character a reviewer use when writing the review, punct-count represents the total number of punctuation that found in each review, and emojis counts the total number of emojis in each review. Also, we have used statistical analysis on reviewers’ behaviors by applying “group by” function, that gets the number of fake or real reviews by each reviewer that are written on a certain date and on each hotel. All these features are taken into consideration to see the effect of the users behaviors on the performance of the classifiers.

5.4 Creating a Model:

We have considered fake review dataset used in our base paper which contains 4 columns namely category, rating, label and text and has 40433 rows, we preprocessed our dataset and created a model of LSTM with 12 epoches.

5.5 Display of Output:

Output displays the accuracy our model has achieved in testing after training.

RoBERTa Architecture:

RoBERTa stands for Robustly Optimized BERT Pre-training Approach. It was presented by researchers at Facebook and Washington University. The goal of this paper was to optimize the training of BERT architecture in order to take lesser time during pre-training

RoBERTa has almost similar architecture as compare to BERT, but in order to improve the results on BERT architecture, the authors made some simple design changes in its architecture and training procedure.

We have used Pytorch for the coding process of this algorithm.We imported the fake reviews dataset and Spliced it into training and testing sets in 70:30 ratio respectively. We have used the 'RobertaTokeniser' from pretrained 'roberta-base' to preprocess the text.We trained the model using the 'RobertaForSequenceClassification' from pretrained 'roberta-base', and validated the model and evaluated it through various evaluation metrics.

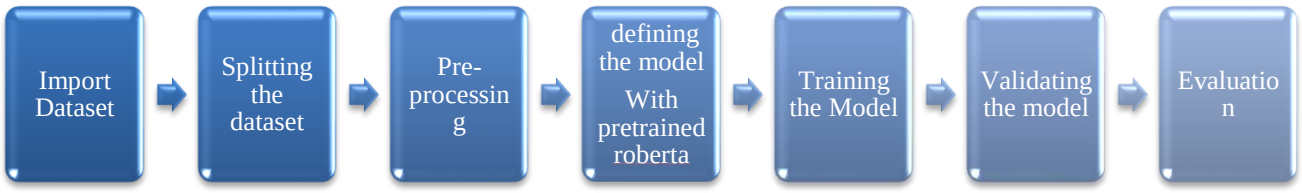


Figure 7: RoBERTa Arhcitecture

LSTM Architecture:

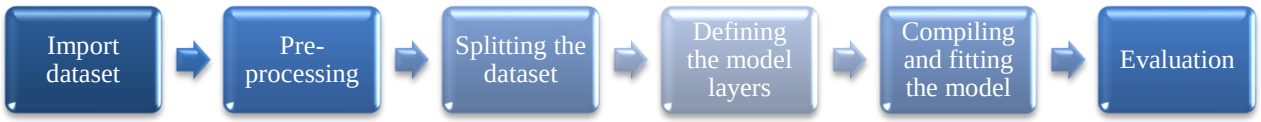


Figure: 6 LSTM Arhcitecture

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). This characteristic makes LSTM networks ideal for processing and predicting data. For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare.

We first import the fake review dataset and preprocess the dataset using the RE library. In the cleaning of text the following steps have been performed:

- Stopwords removal
- Bad symbols are removed
- Special characters have been replaced by a space
- HTML decoding using Beautiful soup

We split the dataset into training and tesing datasets in 70 :30 ratio. The we converted the text into word embeddings using the ‘glove.6B.100d.txt’ from kaggle.We then defined the sequential model and it’s layers using the Keras Library, then we compiled and fitted the model into our LSTM algorithms and evaluated the model through different parameters.

Bi-LSTM ARCHITECTURE:

A Bidirectional LSTM, or Bi-LSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. Bi-LSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence).

A bidirectional LSTM layer learns bidirectional long-term dependencies between time steps of time series or sequence data. These dependencies can be useful when you want the network to learn from the complete time series at each time step.

The training process of Bi-LSTM is similar to the LSTM expect that it has 2 layers of LSTM in the sequential model one learns from front and other from back.Additionally it has an attention layer for better and deep learning of the text.

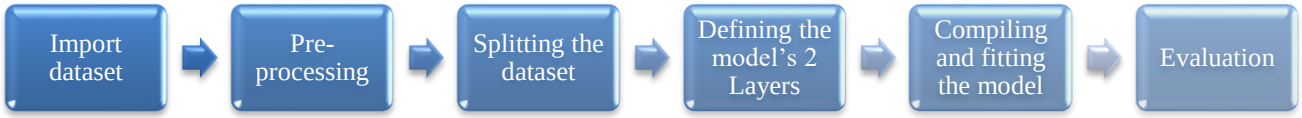


Figure: 8 Bi-LSTM Arhcitecture

NB-LOGISTIC REGRESSION ARCHITECTURE:

NBLR was introduced by Sida Wang and Chris Manning in the paper Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In this research, we use sklearn's logistic regression, rather than SVM, although in practice the two are nearly identical (sklearn uses the liblinear library behind the scenes).

Here we preprocessed the text using the RE library, String Library and TfidfVectoriser for n-gram. We defined the basic naïve bayes feature using the count vectors formula to be used in the log count ratio for the Logistic Model. Then we evaluated the model with various parameters.

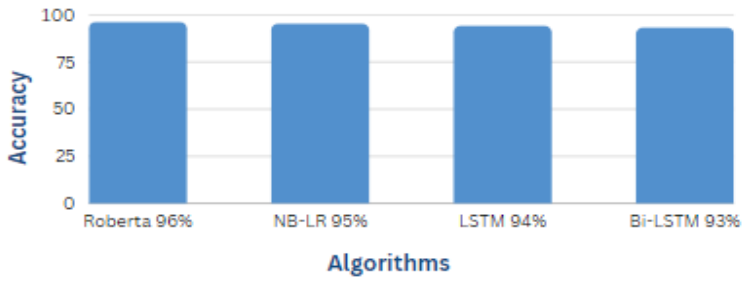
Figure: 9 NB – LR Arhcitecture

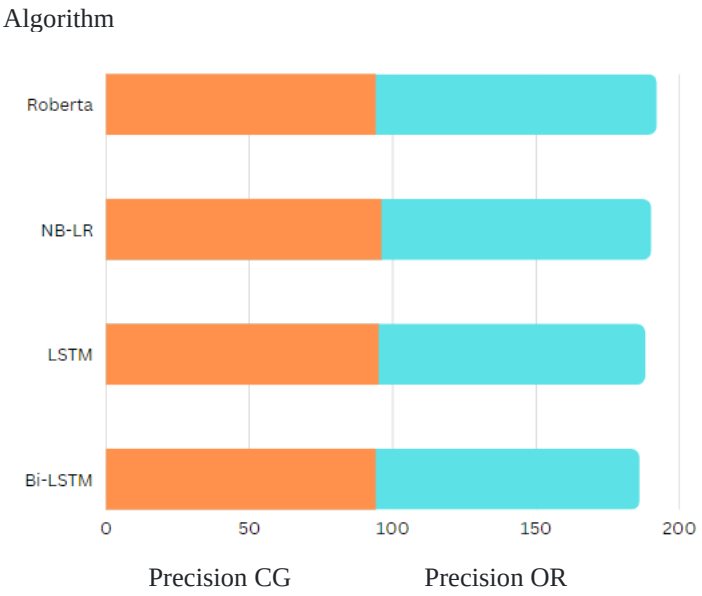
6. RESULTS

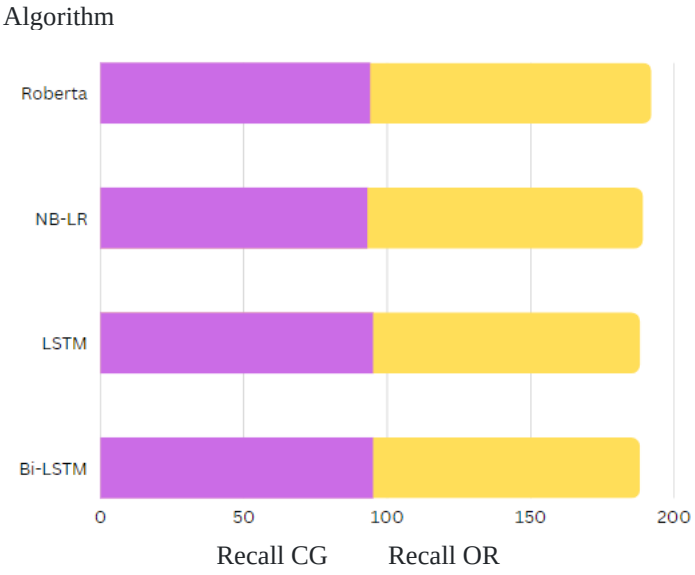
Among all the algorithms used the transformers model Roberta has outperformed others. It's accuracy is 96% followed by the combination model of Naïve Bayes-Logistic Regression with accuracy 95%. Though Bi-LSTM is a better algorithm and successor of LSTM it's very slow compared to LSTM and in some cases it doesn't give better results than later.In our case LSTM gave better results with accuracy 94% and Bi-LSTM with accuracy 93%

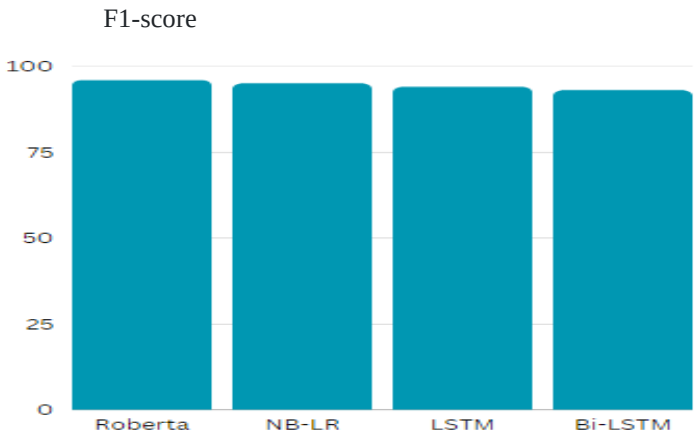
Algorithm	Accuracy in %	Precision in %	Recall in %	F1-Score in %
Roberta	96	CG - 94 OR - 98	CG - 94 OR - 98	CG - 96 OR - 96
NaiveBayes-Logistic Regression	95	CG - 96 OR - 94	CG - 93 OR - 96	CG - 95 OR - 95
LSTM	94	CG - 95 OR - 93	CG - 96 OR - 93	CG - 94 OR - 94
Bi-LSTM	93	CG - 94 OR - 92	CG - 95 OR - 93	CG - 93 OR - 93

Comparison:




Algorithm
Roberta
NB-LR
LSTM
Bi-LSTM
0
50
100
150
200
Precision CG
Precision OR


Algorithm
Roberta
NB-LR
LSTM
Bi-LSTM
0
50
100
150
200
Recall CG
Recall OR


F1-score
100
75
50
25
0
Roberta
NB-LR
LSTM
Bi-LSTM

7. CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Detection of fake reviews is a problem for researchers, e-commerce sites, and firms engaged in online business. Our results indicate that current text generation methods yield fake reviews that appear so realistic that it is challenging for a human to detect them. Fortunately, machine learning classifiers do much better in this regard, with almost perfect accuracy in detecting reviews generated by other machines. In our work, we have found that the Roberta is a miracle algorithm developed by google for spam detection.

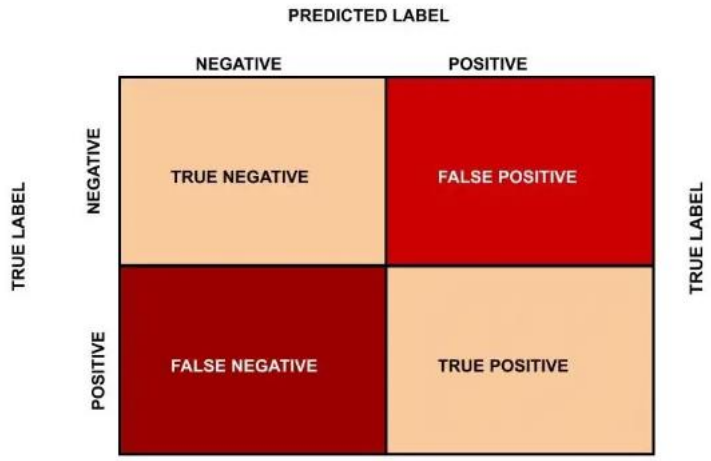
7.2 FUTURE WORK

- Implementing feature extraction and feature engineering for models.
- Implementing ensemble techniques.
- Implement a front end for the web application.
- Try to differentiate spam and non-spam reviews on yelp hotel reviews dataset.
- Using GPT-3 for creation of fake reviews for better training of alogrithms.

8. APPENDICES

8.1 FORMULAE:

Confusion Matrix:



Accuracy = (TN+TP) / (TP+TN+FP+FN)

Precision = (TP) / (TP+FP)

Recall = (TP) / (TP+FN)

F1-Score = 2*(precision*recall) / (precision+recall)

8.2 CODE :

GitHub Link

```
import pandas as pd
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import RobertaForSequenceClassification, RobertaTokenizer
from sklearn.model_selection import train_test_split
```

```
!pip install torch

from torch import cuda
device = 'cuda' if cuda.is_available() else 'cpu'
device

encoded_label_dict = {"CG" : 0, "OR" : 1}
def encode_label(x):
 return encoded_label_dict.get(x,-1)

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

df = pd.read_csv("fake reviews dataset.csv")

df["target"] = df["label"].apply(lambda x: encode_label(x))
model_name = "roberta-base"
MAX_LEN = 256
TRAIN_BATCH_SIZE = 8
VALID_BATCH_SIZE = 8
EPOCHS = 1
LEARNING_RATE = 1e-05
tokenizer = RobertaTokenizer.from_pretrained(model_name)

class Triage(Dataset):
    def __init__(self, dataframe, tokenizer, max_len):
        self.len = len(dataframe)
        self.data = dataframe
        self.tokenizer = tokenizer
```

```
  self.max_len = max_len

def __getitem__(self, index):
        title = str(self.data.text[index])
        title = " ".join(title.split())
        inputs = self.tokenizer.encode_plus(
            title,
            None,
            add_special_tokens=True,
            max_length=self.max_len,
            pad_to_max_length=True,
            return_token_type_ids=True,
            truncation=True
        )
        ids = inputs['input_ids']
        mask = inputs['attention_mask']

        return {
            'ids': torch.tensor(ids, dtype=torch.long),
            'mask': torch.tensor(mask, dtype=torch.long),
            'targets': torch.tensor(self.data.target[index], dtype=torch.long)
        }

    def __len__(self):
        return self.len

# Creating the dataset and dataloader
train_dataset, valid_dataset = train_test_split(df, test_size=0.2, shuffle=True,
```

```
stratify=None, random_state=2021)
train_dataset = train_dataset.reset_index(drop=True)
valid_dataset = valid_dataset.reset_index(drop=True)

print("FULL Dataset: {}".format(df.shape))
print("TRAIN Dataset: {}".format(train_dataset.shape))
print("VALID Dataset: {}".format(valid_dataset.shape))

training_set = Triage(train_dataset, tokenizer, MAX_LEN)
testing_set = Triage(valid_dataset, tokenizer, MAX_LEN)

train_params = {'batch_size': TRAIN_BATCH_SIZE,
                'shuffle': True,
                'num_workers': 0
                }

valid_params = {'batch_size': VALID_BATCH_SIZE,
                'shuffle': True,
                'num_workers': 0
                }

training_loader = DataLoader(training_set, **train_params)
testing_loader = DataLoader(testing_set, **valid_params)

model = RobertaForSequenceClassification.from_pretrained(model_name)
model.to(device)

optimizer = torch.optim.Adam(params=model.parameters(), lr=LEARNING_RATE)
```

```
def calcuate_accu(big_idx, targets):
    n_correct = (big_idx==targets).sum().item()
    return n_correct
def train(epoch):
    tr_loss = 0


  n_correct = 0
    nb_tr_steps = 0
    nb_tr_examples = 0
    model.train()
    for _,data in enumerate(training_loader, 0):
        ids = data['ids'].to(device, dtype = torch.long)
        mask = data['mask'].to(device, dtype = torch.long)
        targets = data['targets'].to(device, dtype = torch.long)

        optimizer.zero_grad()
        outputs = model(ids, attention_mask=mask, labels=targets)
        loss = outputs.loss
        logits = outputs.logits
        tr_loss += loss
        big_val, big_idx = torch.max(logits, dim=1)
        n_correct += calcuate_accu(big_idx, targets)

        nb_tr_steps += 1
        nb_tr_examples+=targets.size(0)

        if _!=0 and _%100==0:
            loss_step = tr_loss/nb_tr_steps
            accu_step = (n_correct*100)/nb_tr_examples
```

```
            print(f"Training Loss per 100 steps: {loss_step}")
            print(f"Training Accuracy per 100 steps: {accu_step}")


        loss.backward()
        optimizer.step()
    print(f'The Total Accuracy for Epoch {epoch}: {(n_correct*100)/nb_tr_examples}')




epoch_loss = tr_loss/nb_tr_steps
    epoch_accu = (n_correct*100)/nb_tr_examples
    print(f"Training Loss Epoch: {epoch_loss}")
    print(f"Training Accuracy Epoch: {epoch_accu}")
   return
def valid(model, testing_loader):
    model.eval()
    n_correct = 0
    n_wrong = 0
    total = 0
    tr_loss = 0
    nb_tr_steps = 0
    nb_tr_examples = 0
    with torch.no_grad():
        for _, data in enumerate(testing_loader, 0):
            ids = data['ids'].to(device, dtype = torch.long)
            mask = data['mask'].to(device, dtype = torch.long)
            targets = data['targets'].to(device, dtype = torch.long)
            outputs = model(ids, attention_mask=mask, labels=targets)
            loss = outputs.loss
```

```
        logits = outputs.logits
        tr_loss += loss
        big_val, big_idx = torch.max(logits, dim=1)
        n_correct += calcuate_accu(big_idx, targets)

        nb_tr_steps += 1
        nb_tr_examples+=targets.size(0)

 if _!=0 and _%100==0:
            loss_step = tr_loss/nb_tr_steps
            accu_step = (n_correct*100)/nb_tr_examples
            print(f"Validation Loss per 100 steps: {loss_step}")
            print(f"Validation Accuracy per 100 steps: {accu_step}")
    epoch_loss = tr_loss/nb_tr_steps
    epoch_accu = (n_correct*100)/nb_tr_examples
    print(f"Validation Loss Epoch: {epoch_loss}")
    print(f"Validation Accuracy Epoch: {epoch_accu}")

    return epoch_accu

tokenizer.pad_token_id

for epoch in range(EPOCHS):
    train(epoch)

acc = valid(model, testing_loader)
print("Accuracy on validation data = %0.2f%%" % acc)

# Save the model
```

```
output_model_file = 'ft-roberta-amazonreviews.pt'
model_to_save = model
torch.save(model_to_save, output_model_file)
print('All files saved')

from transformers import RobertaForSequenceClassification, RobertaTokenizer
import torch



model_name = "roberta-base"
tokenizer = RobertaTokenizer.from_pretrained(model_name)

model = torch.load('ft-roberta-amazonreviews.pt')

preds, preds_probas = [],[]
for i, row in valid_dataset.iterrows():
    query = row["text"]
    pred = predict(query,model,tokenizer)
    preds_probas.append(pred)
    if pred >= 0.5:
        preds.append(1)
    else:
        preds.append(0)

from sklearn.metrics import confusion_matrix
y_true = valid_dataset.target.values
y_pred = preds
confusion_matrix(y_true,y_pred)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report
acc = accuracy_score(y_true,y_pred)
precision = precision_score(y_true,y_pred)
recall = recall_score(y_true,y_pred)

print(f"Accuracy: {acc*100}; Precision:{precision*100}; Recall:{recall*100}")

print(classification_report(y_true, y_pred, target_names=["CG","OR"]))
```

9. REFERENCES

References:

[1] Base Paper - Gupta, P., Gandhi, S., & Chakravarthi, B. R. (2021, December). Leveraging transfer learning techniques-bert, roberta, albert and distilbert for fake review detection. In Forum for Information Retrieval Evaluation (pp. 75-82).

[2] Gutierrez-Espinoza, L., Abri, F., Namin, A. S., Jones, K. S., & Sears, D. R. (2020, July). Ensemble learning for detecting fake reviews. In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1320-1325). IEEE.Hirlekar, V. V., & Kumar, A. (2020, June). Natural language processing based online fake news detection challenges–A detailed review. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)* (pp. 748-754). IEEE.

[3] Hou, J., & Zhu, A. (2021). Fake online review recognition algorithm and optimisation research based on deep learning. *Applied Mathematics and Nonlinear Sciences*, *7*(2), 861-874.

[4] Briskilal, J., & Subalalitha, C. N. (2022). An ensemble model for classifying idioms and literal texts using BERT and RoBERTa. Information Processing & Management, 59(1), 102756.

[5] Taşağal, K., & Uçar, Ö. (2018). Detection of Fake User Reviews with Deep Learning. *International Journal of Research in Engineering and Applied Sciences (IJREAS)*, *8*(12)

[6] Crawford, M Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Al Najada, H. (2015). Survey of review spam detection using machine learning techniques. *Journal of Big Data*, *2*(1), 1-24.

[7] Trivedi, S. K. (2016, September). A study of machine learning classifiers for spam detection. In 2016 4th international symposium on computational and business intelligence (ISCBI) (pp. 176-180). IEEE.

[8] Makkar, A., Garg, S., Kumar, N., Hossain, M. S., Ghoneim, A., & Alrashoud, M. (2020). An efficient spam detection technique for IoT devices using machine learning. IEEE Transactions on Industrial Informatics, 17(2), 903-912.

[9] Al-Adhaileh, M. H., & Alsaade, F. W. (2022). Detecting and Analysing Fake Opinions Using Artificial Intelligence Algorithms. Intelligent Automation & Soft Computing, 32

[10] Shirani-Mehr, H. (2013). SMS spam detection using machine learning approach. unpublished) http://cs229. stanford. edu/proj2013/Shir aniMeh r-SMSSpamDetectionUsingMachineLearningApproach. Pdf.

[11] Kumar, A., Gopal, R. D., Shankar, R., & Tan, K. H. (2022). Fraudulent review detection model focusing on emotional expressions and explicit aspects: investigating the potential of feature engineering. Decision Support Systems, 155, 113728.

10. GLOSSARY

LSTM: LSTMs are predominately used to learn, process, and classify sequential data because these networks can learn long-term dependencies between time steps of data. Common LSTM applications include sentiment analysis, language modeling, speech recognition, and video analysis.

BiLSTM: A bidirectional LSTM (BiLSTM) layer learns bidirectional long-term dependencies between time steps of time series or sequence data. These dependencies can be useful when you want the network to learn from the complete time series at each time step.

BERT: BERT is an open source machine learning framework for natural language processing (NLP). BERT is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context.

ROBERTa: RoBERTa, which was implemented in PyTorch, modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates.

NaiveBayes - Logistic Regression: The combining model of NB and LR introduces the basic naïve bayes feature in the Logistic regression.

Pandas: Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series.

NumPy: It is a library for the Python Programming Language which is used to support large multi-dimensional arrays and matrices along with a large collection of high level

mathematical functions to operate on the arrays. Since an image is basically a huge matrix of pixel values, numpy is a great library for preprocessing the images.

Matplotlib: It is a Plotting library for the Python Programming Language. It is a static library which is used to create static, animated and interactive visualizations.This project uses Matplotlib to verify the results of preprocessing.

Neural Network: A set of algorithms that aims to recognize the underlying patterns in data is called a Neural Network. It comprises of Artificial Neurons or Nodes which try to mimic the way a human brain operates to identify patterns and derive meaningful conclusions from them.

RNN: A recurrent neural network is a type of artificial neural network commonly used in speech recognition and natural language processing. Recurrent neural networks recognize data's sequential characteristics and use patterns to predict the next likely scenario.

Classification: In Machine Learning, Classification refers to a predictive modeling problem where a class label is predicted for an for a given example of data. In this project, Class labels are the Characters in the dataset. The label is assigned to the input text data.

TensorFlow: The TensorFlow platform helps you implement best practices for data automation, model tracking, performance monitoring, and model retraining. Using production-level tools to automate and track model training over the lifetime of a product, service, or business process is critical to success.

NLTK: NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to preprocess text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

PYTORCH: PyTorch is a fully featured framework for building deep learning models, which is a type of machine learning that's commonly used in applications like image recognition and language processing. Written in Python, it's relatively easy for most machine learning developers to learn and use. PyTorch is distinctive for its excellent support for GPUs and its use of reverse-mode auto-differentiation, which enables computation graphs to be modified on the fly. This makes it a popular choice for fast experimentation and prototyping.

TRANSFORMERS: Transformers also known as pytorch-transformers and pytorch-pretrained-bert provides general-purpose architectures like XLM, DistilBert, XLNet, BERT, GPT-2, RoBERTa, etc for Natural Language Understanding(NLU) and Natural Language Generation (NLG) with over 30+ pre-trained models in 100+ languages and deep interoperability between Jax, PyTorch, and TensorFlow.

PICKLE: The pickle module implements binary protocols for serializing and de-serializing a Python object structure. *"Pickling"* is the process whereby a Python object hierarchy is converted into a byte stream, and *"unpickling"* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

RE LIBRARY: A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

KERAS: Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

BS4: Beautiful Soup (bs4) is a python web scraping library for pulling the data from web pages, documents, HTML, and XML files.

GENSIUM: Gensium is a free open-source Python library for representing documents as semantic vectors, as efficiently (computer-wise) and painlessly (human-wise) as

possible.

LOGGING: Logging is used to track events that happen when an application runs. Logging calls are added to application code to record or log the events and errors that occur during program execution. In Python, the logging module is used to log such events and errors.